공학석사학위논문

모바일 엣지 컴퓨팅 환경에서의 연산 오프로딩을 위한 이동경로 예측

Mobility Prediction for Computation Offloading in the Mobile Edge Computing Environment

2019년 8월

서울대학교 대학원

전기정보공학부

이 현 재

공학석사학위논문

모바일 엣지 컴퓨팅 환경에서의 연산 오프로딩을 위한 이동경로 예측

Mobility Prediction for Computation Offloading in the Mobile Edge Computing Environment

2019년 8월

서울대학교 대학원

전기정보공학부

이 현 재

모바일 엣지 컴퓨팅 환경에서의 연산 오프로딩을 위한 이동경로 예측

Mobility Prediction for Computation Offloading in the Mobile Edge Computing Environment

지도교수 문 수 묵

이 논문을 공학석사 학위논문으로 제출함

2019년 8월

서울대학교 대학원

전기정보공학부

이 현 재

이현재의 공학석사 학위 논문을 인준함

2019년 8월

위 원 장: 백 윤 흥 (인)

부위원장: 문 수 묵 (인)

위 원: 윤 성 로 (인)

초록

모바일 기기의 부족한 컴퓨팅 파워와 배터리 용량으로 인해 연산량이 많은 어플리케이션을 모바일 기기에서 수행하기에는 무리가 있다. 이에 대한 해결책 중 하나는 중앙 클라우드 서버로 연산을 오프로딩 하여 서버가 연산을 대신 수행하도록 하는 것이다. 하지만 중앙 클라우드 서버를 이용하게 되면 데이터를 주고 받으며 발생하는 시간 지연(latency)이 커지게 된다. 이러한 시간 지연은 실시간으로 연산을 해야 하는 어플리케이션의 수행을 어렵게 만든다.

중앙 클라우드 서버의 시간 지연 문제를 해결하기 위해 모바일 엣지 컴퓨팅(MEC)이라는 새로운 패러다임이 등장했다. 모바일 엣지 컴퓨팅에서는 모바일 기기와 가까운 엣지 서버로 연산을 오프로딩 하기 때문에 전통적인 클라우드 컴퓨팅에서 중앙 서버를 이용하기 때문에 발생했던 시간 지연을 줄일 수 있다. 하지만 모바일 엣지 컴퓨팅 환경에서는 모바일 유저가 이동하면서 오프로딩하는 서버가 계속해서 바뀌게 된다. 서버가 바뀔 때마다 오프로딩에 필요한 데이터를 다시 전송해야 하고, 데이터를 전송하는 동안은 서버를 이용할 수 없기 때문에 성능 저하가 발생한다.

본 논문에서는 모바일 유저의 과거 위치정보를 기반으로 이동경로를 예측하여 다음에 연결될 엣지 서버로 연산 오프로딩을 위한 데이터를 미리 보내 전체적인 오프로딩 성능을 향상시킬 수 있도록 이동경로를 예측하는 모델을 제시한다. 이동경로 예측 모델은 Markov Model, Support Vector Regression, Recurrent Neural Network를 사용하여 구현하였고, 시뮬레이션을 통해 이동경로 예측 모델이 모바일 엣지 컴퓨팅 환경에서의 연산 오프로딩 성능을 향상시키는 것을 확인하였다.

주요어: 이동경로 예측, 연산 오프로딩, 모바일 엣지 컴퓨팅

학 번: 2017-28906

목차

표 목차

그림 목차

제 1 장 서 론

스마트폰과 태블릿 컴퓨터같은 모바일 기기들의 사용량이 나날이 늘어나고 있다. 최근에는 기술의 발전으로 모바일 기기들의 성능이 계속해서 향상되고 있으며, 이에 따라 사람들이 모바일 기기에서 기대하는 성능도 같이 증가하고 있다. 하지만 이러한 성능 향상에도 불구하고 모바일 기기의 컴퓨팅 파워와 배터리 용량은 복잡한 연산을 수행하기에 여전히 부족하다. 특히 머신러닝이나 딥러닝처럼 복잡하고 많은 연산을 요구하는 어플리케이션을 모바일 기기에서 수행하기는 쉽지 않다.

이러한 문제점을 해결하기 위해 모바일 클라우드 컴퓨팅(Mobile Cloud Computing)이라는 개념이 등장했다. 모바일 클라우드 컴퓨팅은 뛰어난 연산 능력을 가진 클라우드 서버가 모바일 기기를 대신해 복잡한 연산을 수행하여 모바일 기기의 부족한 성능을 극복할 수 있게 해준다. 하지만 모바일 클라우드 컴퓨팅은 중앙화된 클라우드 서버를 이용하기 때문에 데이터를 주고받을 때 latency가 길다는 단점이 있다. 모바일 클라우드 컴퓨팅의 high latency는 실시간으로 빠르게 연산을 해야하는 증강 현실(Augmented Reality)이나 클라우드 게이밍(Cloud Gaming)같은 어플리케이션의 수행을 어렵게 만든다.

모바일 클라우드 컴퓨팅의 high latency 문제에 대한 해결책으로 모바일 엣지 컴퓨팅(Mobile Edge Computing)이라는 기술이 제안되었다 [1][2]. 모바일 엣지 컴퓨팅은 모바일 클라우드 컴퓨팅이 중앙화된 클라우드 서버를 이용하면서 발생했던 high latency 문제를 사용자와 지리적으로 가까운 엣지 서버를 이용함으로써 해결한다. 엣지 서버는 클라우드 서버와는 달리 여러 장소에 분산되어 있기 때문에 모바일 기기는 자신으로부터 가장 가까이 있는 엣지 서버를 통해 low latency로 컴퓨팅 파워를 제공받을 수 있다.

모바일 기기가 서버의 컴퓨팅 파워를 이용하기 위해서는 연산을 서버로 오프로딩 하기 전에 연산을 위한 데이터가 미리 서버에 저장되어 있어야 한다. 중앙 클라

우드 서버를 이용하는 경우에는 오프로딩에 필요한 데이터들을 미리 서버로 전송해두어 오프로딩을 위한 환경을 미리 구축해놓고 모바일 기기가 계속해서 서버를 이용할 수 있다. 하지만 엣지 서버를 이용하는 경우에는 그렇지 않다. 모바일 엣지 컴퓨팅 환경에서는 모바일 유저가 이동하면서 사용할 수 있는 엣지 서버가 계속해서 바뀌기 때문에 연산 오프로딩을 위한 환경을 서버에 미리 구축해놓기 쉽지 않다. 따라서, 모바일 유저가 새로운 엣지 서버에 접속할 때마다 연산 오프로딩을 위한 데이터를 서버로 전송해야 하고, 데이터가 다 전송되기 전까지는 엣지 서버를 이용할 수 없기 때문에 전체적인 성능 저하가 발생한다.

본 논문에서는 모바일 엣지 컴퓨팅 환경에서 발생하는 연산 오프로딩 성능 저하를 개선할 수 있는 방식을 제안한다. 오프로딩 성능을 향상시키기 위해 제안하는 방식은 모바일 유저의 이동경로를 예측해 다음에 연결될 엣지 서버를 예측하여 오프로딩에 필요한 데이터를 미리 전송하는 것이다.

이동경로를 예측하는 모델은 모바일 유저들의 과거 이동경로 정보를 기반으로 다음 위치를 예측하도록 구현하였다. 이동경로 예측 모델을 통해 모바일 유저의 다음 위치를 예측하고, 예측한 위치로부터 가장 가까운 엣지 서버로 오프로딩을 위한 데이터를 미리 전송한다. 이를 통해 모바일 기기가 데이터를 전송하는 동안 서버를 이용하지 못하는 시간을 줄여 성능 저하를 줄일 수 있도록 하였다.

이후 논문은 다음과 같은 순서로 구성된다. 제 2장에서는 모바일 엣지 컴퓨팅의 개념과 엣지 컴퓨팅 환경에서 발생하는 연산 오프로딩의 성능 저하를 다룬다. 이어지는 제 3장에서는 이동경로 예측 문제를 정의하고 문제를 해결하기 위한 모델에 대해 설명한다. 제 4장에서는 이동경로 예측 모델의 성능을 측정한 결과와, 예측 모델이 모바일 엣지 컴퓨팅 환경에서의 오프로딩 성능에 주는 효과를 확인하기 위해 수행한 시뮬레이션 결과를 다룬다. 마지막으로 제 5장에서 결론으로 마무리한다.

제 2 장 모바일 엣지 컴퓨팅

2.1 모바일 엣지 컴퓨팅 환경

모바일 엣지 컴퓨팅 환경은 일반적인 클라우드 컴퓨팅 환경과는 다른 특징을 가진다. 모바일 엣지 컴퓨팅이 클라우드 컴퓨팅 환경과 다른 가장 큰 특징 중 하나는 컴퓨팅 파워를 제공하는 서버가 여러 장소에 분포되어 있다는 것이다. 따라서 모바일 엣지 컴퓨팅 환경에서 모바일 유저가 서버의 컴퓨팅 파워를 이용하고자 하는 경우, 클라우드 서버보다 지리적으로 유저와 훨씬 가까운 엣지 서버를 이용할 수 있기 때문에 low latency로 컴퓨팅 파워를 제공받을 수 있다.

모바일 엣지 컴퓨팅 환경에서 엣지 서버를 연산 오프로딩 서버로 이용하면 latency가 작기 때문에 기존 클라우드 서버가 가지고 있던 high latency 문제가 해결되어 증강 현실 어플리케이션이나 클라우드 게이밍처럼 low latency를 요구하는 어플리케이션을 수행할 수 있게 된다.

지리적으로 가까운 엣지 서버를 이용하면 low latency로 컴퓨팅 파워를 제공받을 수 있다는 장점이 있지만, low latency를 유지하기 위해 발생하는 문제점도 가지고 있다. 계속해서 움직이는 모바일 유저가 low latency를 유지하려면 지리적으로 가장 가까운 서버를 오프로딩 서버로 이용해야 한다. 하지만 서버로 연산을 오프로딩 하기 위해서는 서버가 오프로딩을 위해 필요한 데이터를 갖고 있어야 하는데, 모바일 유저와 가장 가까운 서버가 바뀌게 되면 그 서버는 오프로딩을 위한 데이터를 갖고 있지 않기 때문에 오프로딩을 위해 필요한 데이터를 이전 서버에서 다음 서버로 넘겨주는 핸드오프(handoff) 과정이 필요하다(그림 1).

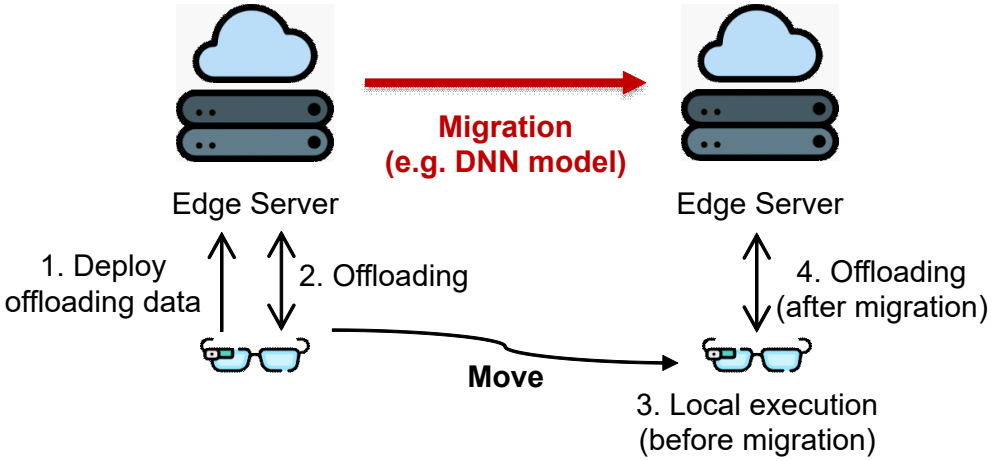

그림 1. 모바일 엣지 컴퓨팅 환경에서의 핸드오프

2.2 핸드오프시 연산 오프로딩의 성능 저하

핸드오프는 엣지 컴퓨팅 환경에서 모바일 유저가 low latency로 컴퓨팅 파워를 제공하기 위해 반드시 필요한 작업이다. 하지만 핸드오프 과정 중에는 서버의 컴퓨팅 파워를 제공받을 수 없어 그 시간동안 오프로딩을 할 수 없는 문제가 발생한다. 또한, 핸드오프는 이전 서버에서 다음 서버로 데이터를 전송하는 과정이 필요한데, 만약 딥러닝 모델 같이 매우 큰 데이터를 전송해야 하는 경우 핸드오프 시간이 길어지게 되어 더 오랜 시간동안 서버를 이용할 수 없게 된다.

2.3 제안하는 해결 방식

모바일 엣지 컴퓨팅 환경에서의 오프로딩 성능 저하는 핸프오프 과정에서 서버의 컴퓨팅 파워를 이용할 수 없기 때문에 발생한다. 따라서 만약 핸드오프를 모바일 유저가 다른 엣지 서버로 이동하기 전에 미리 하게 된다면, 서버를 이용할 수 없는 시간을 줄일 수 있을 것이다.

본 연구에서는 모바일 유저의 이동경로를 예측하는 모델을 구현하고, 이를 통해 유저가 다음에 오프로딩 서버로 사용할 엣지 서버를 예측해 데이터를 미리 보냄으로써 모바일 엣지 컴퓨팅 환경에서 오프로딩 성능을 향상시키고자 한다.

제 3 장 이동경로 예측

3.1 문제 정의

정의 1 *location*(위치 정보): 모바일 유저의 *location*은 경도 (longitude)와 위도 (latitude)의 쌍으로 나타낼 수 있다. 본 연구에서 유저의 위치는 고정된 장소를 기준으로 얼마만큼 떨어져 있는지로 나타낸다. 기준점으로부터 경도 방향으로 떨어진 거리를 x, 위도 방향으로 떨어진 거리를 y라고 하면 모바일 유저의 *location*은 (x, y)로 나타낼 수 있다.

정의 2 *time interval*(시간 간격): 모바일 유저의 위치 정보를 수집할 때 일정 시간 간격 t를 두고 수집하게 된다. 이동경로 예측 모델은 t초 간격으로 수집된 과거 n개의 위치 정보를 이용하여 사용자의 다음 t초 후 위치를 예측한다.

정의 3 *trajectory*(과거 이동경로): 본 연구에서의 이동경로 예측은 사용자의 과거 위치 정보를 기반으로 한다. 시간 T에서 과거 n개의 위치 정보를 기반으로 다음 위치를 예측한다면, 사용자의 과거 이동경로는 다음과 같이 나타낼 수 있다.

$$trajectory = (x_{T-(n+1)\times t}, y_{T-(n+1)\times t}), (x_{T-(n+2)\times t}, y_{T-(n+2)\times t}), ..., (x_T, y_T)$$

이동경로 예측 문제는 사용자의 과거 이동경로가 주어졌을 때, 다음 위치를 예측하는 문제이다. 예측 모델을 **F**라는 함수로 표현하면, 시간 T에서 시간 간격 t로 수집된 과거 이동경로를 기반으로 시간 $T + t$에서의 위치를 예측하는 문제는 다음과 같은 식에서 함수 **F**를 찾아내는 것과 같다.

$$(x_{n+1}, y_{n+1}) = \mathbf{F}((x_{T-(n+1)\times t}, y_{T-(n+1)\times t}), (x_{T-(n+2)\times t}, y_{T-(n+2)\times t}), ..., (x_T, y_T))$$

3.2 이전 연구

3.2.1 Markov-Based Approach

이전 연구[3][4]에서는 사용자의 이동경로를 예측하기 Markov model을 사용한다. 특정 지역을 사각형의 cell로 구분하고 사용자의 과거 위치를 cell에 mapping 시킨 후에, cell의 sequence를 기반으로 다음 cell을 예측한다.

3.2.2 Deep Learning-Based Approach

[5]에서는 Recurrent Neural Network(RNN)를 사용해 모바일 유저의 다음 위치를 예측했다. 이 연구에서 구현한 모델은 과거 몇 개의 위치 정보를 기반으로 모바일 유저의 몇 십분 후의 위치를 예측한다. [6]에서는 [5]와 마찬가지로 RNN을 사용하여 모바일 유저의 위치를 예측하는데, 이 연구에서는 위치 뿐만 아니라 이동수단도 함께 예측한다.

3.3 이동경로 예측 모델

본 연구에서 이동경로 예측을 위한 모델로 Markov Model, Support Vector Regression(SVR), 그리고 Recurrent Neural Network(RNN)를 사용하였다. 구현한 모델들은 기존 연구와는 달리 짧은 시간(<30초) 후의 위치를 예측한다.

3.3.1 Markov Model

이동경로 예측을 위한 Markov Model은 이전 연구[3][7][4]에서 Mobility 예측을 위해 사용했던 일반적인 Markov Model의 확장 모델인 Variable-Order Markov(VOM)를 사용하였다. VOM은 다음 상태를 고정된 길이의 이전 상태만으로 모델링하는 기존 Markov 모델과는 다르게, 상황에 따라 다른 길이의 이전 상태를 통해 다음 상태를 모델링한다. VOM으로 다음 엣지 서버를 예측하기 위해 먼저 모바일 사용자의 위치

정보 (x, y)를 해당 구역의 엣지 서버의 id로 매핑하였다. 엣지 서버의 id로 매핑한 사용자의 이동경로 정보를 기반으로 prediction suffix tree를 만들어 VOM을 구현하였다[8].

3.3.2 Support Vector Regression(SVR)

Support Vector Regression(SVR)은 Support Vector Machine(SVM)의 일종으로 회귀(regression) 문제를 해결하기 위한 모델이다[9]. SVR 모델은 사용자의 과거 이동경로 $Trajectory$를 입력으로 받아 다음 위치 (x, y)를 출력하게 된다. 본 연구에서는 Linear SVR을 예측 모델로 사용하고 epsilon-insensitive hinge loss를 사용하여 모델을 학습시켰으며, 예측 성능이 가장 좋은 모델의 파라미터(epsilon, tolerance)를 실험적으로 결정하였다.

3.3.3 Recurrent Neural Network(RNN)

Recurrent Neural Network(RNN)는 인공 신경망(Artifitial Neural Network)의 일종으로 이전 입력의 출력이 다음 입력으로 들어가는 구조를 가지고 있어, 순차적인(sequential) 데이터를 처리하기 적합한 모델이다. 본 연구에서는 LSTM(Long Short-Term Memory) 셀을 이용하여 RNN 모델을 구현하였다. 사용자의 과거 이동경로 $Trajectory$가 시간 순서로 RNN 모델의 입력으로 들어오면, LSTM 셀은 출력 텐서(output tensor)를 반환하게 된다. LSTM 셀로부터 계산된 출력 텐서는 fully connected layer를 통해 최종적으로 2개의 값을 갖는 텐서를 계산하게 된다. 이렇게 최종적으로 계산된 텐서가 사용자의 다음 위치 (x, y)가 된다. Mean absolute error(MAE)를 손실 함수로 사용하여 RNN 모델을 학습시켰으며, SVR 모델과 마찬가지로 RNN 모델의 파라미터를 실험적으로 결정하였다.

제 4 장 실험 결과 및 분석

4.1 데이터셋

본 논문에서 제시하는 연산 오프로딩의 성능을 측정하기 위해 두 가지 데이터셋을 사용하였다. 각 데이터셋의 80%는 이동경로 예측 모델의 training data로 사용하였고, 20%는 test data로 사용하였다.

4.1.1 KAIST

KAIST 데이터셋은 대학 캠퍼스 내에서 학생들로부터 30초 간격으로 수집된 GPS 데이터이다[10]. 본 연구에서는 KAIST 캠퍼스 주변의 GPS 데이터만을 사용하였다.

4.1.2 Geolife

Geolife 데이터셋은 여러 도시에서 182명의 사용자들로부터 1 5초 간격으로 수집한 GPS 데이터이다[11][12][13]. 본 연구에서는 중국 베이징시의 지하철 2호선을 포함하는 영역의 GPS 데이터만을 사용하였으며 (위도 : 39.900341 $\sim$ 39.950932, 경도 : 116.353370 $\sim$ 116.437765), GPS 데이터 사이의 시간 간격이 일정하도록 pre-processing하였다.

4.2 이동경로 예측 모델의 정확도

이동경로 예측 모델의 성능은 top-n accuracy를 통해 측정하였다. Markov 모델은 모바일 유저의 위치를 엣지 서버의 id로 매핑한 후에 예측하기 때문에 유저의 다음 위치가 아니라 다음 서버를 예측하게 된다. 따라서 Markov 모델의 top-n accuracy

는 모델이 예측하는 서버 중 확률이 가장 높은 n개의 서버 중에서 올바른 예측이 있는 비율이 된다. SVR 모델과 RNN 모델은 모바일 유저의 위치를 예측하기 때문에, top-n accuracy는 모델이 예측한 위치에서 가장 가까운 n개의 서버 중에서 올바른 예측이 있는 비율이 된다.

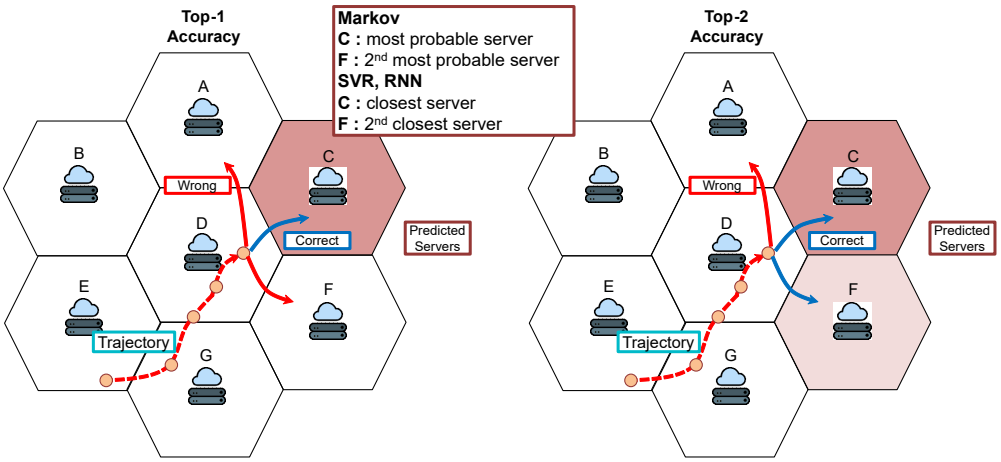

그림 2. 이동경로 예측 모델의 top-1, top-2 accuracy

그림 2는 이동경로 예측 모델의 top-1 accuracy와 top-2 accuracy를 나타낸 것이다. 본 연구에서는 top-1 accuracy와 top-2 accuracy를 측정해 이동경로 예측 모델의 성능을 평가하였다.

4.3 시뮬레이션

본 연구에서 제안하는 이동경로 예측 모델이 모바일 엣지 컴퓨팅 환경에서의 오프로딩 성능에 주는 영향을 확인하기 위해 시뮬레이션을 수행하였다. 시뮬레이션에서는 세 가지 Deep Neural Network(DNN) 모델을 사용하는 image classification 어플리케이션을 사용하였다(표 1). 이 어플리케이션들은 DNN 모델을 통해 계속해

DNN	Number of Layers	Size (MB)
MobileNet	110	16
Inception	312	128
ResNet	245	98

표 1. 시뮬레이션에 사용한 DNN 모델

서 image classification 작업을 수행한다. 만약 엣지 서버로 연산을 오프로딩 할 수 있다면 이 어플리케이션들은 DNN query를 계속해서 서버에게 보내는데, 이전 query가 수행되고 0.5초 후에 다음 query를 보낸 식으로 동작한다. 그리고 오프로딩을 할 수 없는 상황이라면 모바일 디바이스에서 DNN 연산을 수행하게 된다.

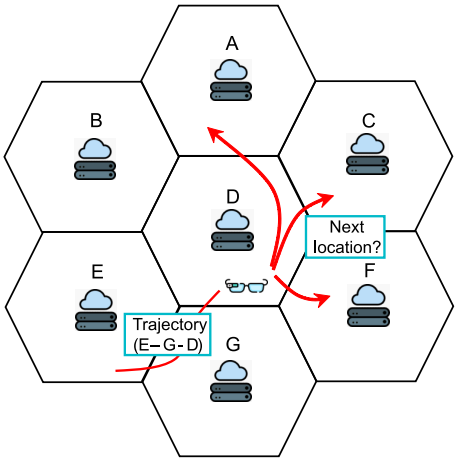

그림 3. 스마트 캠퍼스/시티의 엣지 서버 환경

시뮬레이션에서는 그림 3과 같이 엣지 서버가 균일하게 분포되어 있고 하나의

엣지 서버가 육각형 모양의 지역 (radius: 50m) 을 담당하는 스마트 캠퍼스/시티를 가정하였다. 모바일 유저는 자신이 위치하는 구역을 서비스하는 엣지 서버로 연산을 오프로딩 하게 된다. 예측 모델이 유저가 다음에 이동할 서버를 예측하면 그 서버로 DNN 모델을 전송하게 된다. 모델 전송은 이동경로 예측 모델이 예측한 사용자의 위치를 기준으로 가장 가까운 두 개의 서버로 이루어진다. 서버는 DNN 모델을 전송받으면 일정 시간 동안 모델을 저장하고 있다가 그 시간이 지나면 버리는데, 만약 DNN 모델을 저장하고 있는 도중에 다른 서버가 같은 모델을 전송하려고 한다면 모델 전송은 하지 않고 모델을 버리는 시간을 초기화 시킨다.

시뮬레이션은 실제 환경에서 측정한 결과 (모델 전송 시간, DNN 수행 시간 등)를 바탕으로 수행하였다. 모바일 유저(클라이언트)로는 임베디드 보드 (Arm big.Little CPU 2.0/1.5GHz, 2GB RAM)를, 엣지 서버로는 데스크탑 PC (Intel quad-core CPU (i7-7700), Titan Xp GPU, 32GB RAM)를 사용하였으며, 네트워크는 Wi-Fi (Downlink: 50Mbps, Uplink: 35Mbps)를 사용하였다.

시뮬레이션을 통해 이동경로 예측 모델을 사용하면 얼마나 더 많은 오프로딩 query를 수행할 수 있는지와 사용자가 다른 이동했을 때 DNN 모델이 그 서버로 이미 전송이 완료된 비율 (hit ratio) 을 측정하였다. 오프로딩 query는 사용자가 한 엣지 서버에 머물면서 요청한 static query와 다른 서버로 이동했을 때 요청한 dynamic query로 구분하였다.

4.4 실험 결과

4.4.1 Time interval 및 trajectory 길이

이동경로 예측 모델의 성능은 *time interval*(t)과 과거 *Trajectory*의 길이(n) 에 따라 달라진다. 적절한 t와 n을 설정하는 것이 예측 모델의 성능에 중요한 영향을 끼친다. 따라서 Geolife 데이터셋으로 t와 n을 변화시켜가며 SVR 모델의 예측 성능을 측정하고, 그 결과를 기반으로 t와 n을 결정하였다.

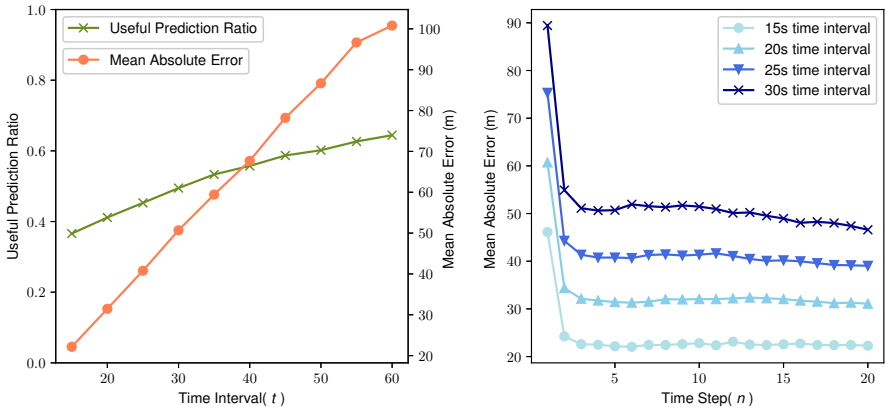

그림 4. 왼쪽: *time interval*(t) 에 따른 useful prediction ratio와 예측 오차, 오른쪽: *trajectory* 길이(n) 에 따른 예측 오차

Time interval(t)을 결정하기 위해 t를 변화시켜가며 두 가지를 측정하였다. 하나는 실제 사용자의 위치와 모델이 예측한 위치 사이의 평균 거리 차이(Mean Absolute Error) 이고, 다른 하나는 전체 이동경로에서 실제로 사용자가 다른 서버로 움직인 비율(Useful Prediction Ratio) 이다. 그림 4의 왼쪽은 t를 변화시켰을 때 예측 성능을 나타낸다. t가 커질수록 useful prediction ratio는 증가하지만 예측 오차(MAE) 도 같이 커지는 것을 확인할 수 있다. Useful prediction ratio는 클수록 좋고, 예측 오차는 작을수록 좋기 때문에 적당한 t를 선택해 둘 사이의 균형을 맞추는 것이 중요하다. 따라서 최종적인 t는 useful prediction ratio와 예측 오차의 비율이 가장 높은 20초로 결정하였다.

그림 4의 오른쪽은 과거 *Trajectory* 길이(n) 에 따른 예측 오차를 나타낸 그래프이다. n이 1에서 2로 증가할 때 예측 오차가 급격하게 감소하는 것을 확인할 수 있다. 하지만 n이 5 이상이 되면 예측 오차가 변화하지 않거나 거의 줄어들지 않기 때문에 n을 5로 결정하였다.

그림 5. 이동경로 예측 모델의 정확도(모든 예측)

4.4.2 이동경로 예측 모델 성능

그림 5은 이동경로 예측 모델의 top-1 accuracy와 top-2 accuracy를 나타낸다. KAIST 데이터셋에서는 Markov, SVR, RNN 모델 모두 top-1 accuracy가 80% 이상이고, top-2 accuracy는 90% 이상으로 상당히 높은 결과가 나왔다. Geolife 데이터셋에서는 Markov 모델의 top-1 accuracy와 top-2 accuracy가 각각 44%, 57%가 나왔고, SVR 모델과 RNN 모델의 top-1 accuracy와 top-2 accuracy는 각각 약 67%, 79% 정도가 나왔다.

KAIST, Geolife 두 데이터셋에서 모두 이동경로 예측 모델의 정확도가 높게 나오는 것을 확인할 수 있다. 특히 KAIST 데이터셋에서의 top-2 accuracy는 90% 이상으로 상당히 높은 정확도를 보인다. KAIST 데이터셋은 대학교 캠퍼스에서 학생들이 이동하는 GPS 데이터이므로 이동하는 속도가 빠르지 않아 많은 경우 같은 서버가 서비스하는 지역에 머물게 된다. Geolife 데이터셋은 베이징시에서 사람들이 이동

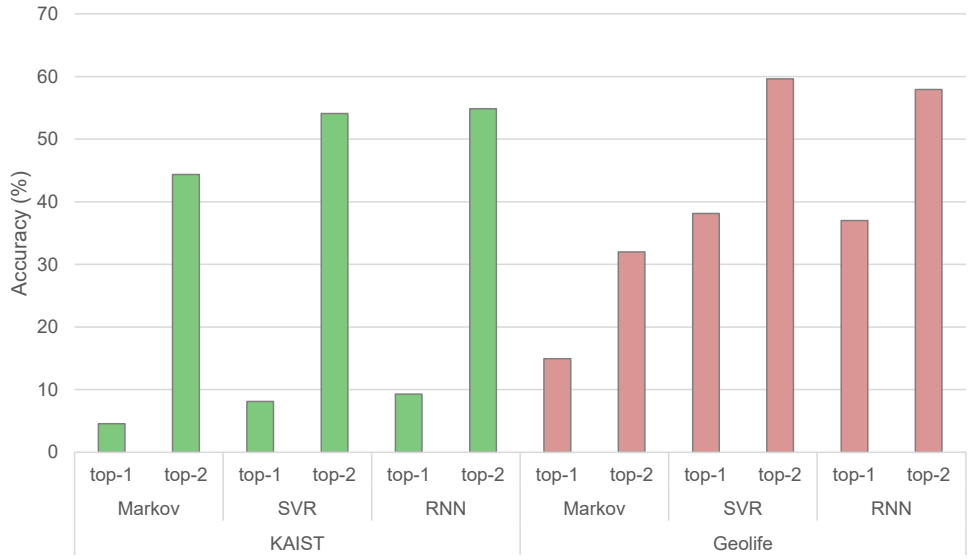

그림 6. 이동경로 예측 모델의 정확도(다른 서버로 이동한 경우만)

하는 GPS 데이터이기 때문에 KAIST 데이터셋보다는 적지만 그래도 꽤 많은 경우 사람들이 같은 서버가 서비스하는 지역에 머물게 된다. 다른 서버로 이동하는 예측보다 기존 서버에 머무를 것이라는 예측의 정확도가 더 높기 때문에 전체적인 정확도가 높아지게 된다.

본 연구에서 사용하는 이동경로 예측 모델은 사용자의 다음 위치를 예측하여 연산 오프로딩을 위한 데이터를 다른 서버에 미리 보내는 것이 목적이다. 따라서 모바일 유저가 기존 서버에 머무르는 것을 잘 예측하는 것보다 다른 서버로 이동할 때 정확하게 예측하는 것이 더 중요하기 때문에 유저가 실제로 다른 서버로 이동하는 경우에만 정확도를 측정했다.

그림 6는 모바일 유저가 실제 다른 서버로 이동하는 경우에만 top-1 accuracy와 top-2 accuracy를 측정한 결과이다. 결과를 보면 top-1 accuracy가 많이 낮아진 것을 확인할 수 있는데, 이는 이동경로 예측 모델은 대부분의 경우 모바일 유저가 기존 서

표 2. 이동경로 예측 모델의 예측 시간

Markov	**SVR**	**RNN**
∼0.02sec	∼0.2sec	∼1sec

버에 머무를 것이라고 예측하는 경우가 많기 때문이다. 마찬가지로 top-2 accuracy 도 모든 경우를 포함해 정확도를 측정했을 때보다 낮아진 것을 확인할 수 있다.

이동경로 예측 모델의 정확도를 보면 SVR 모델과 RNN 모델의 성능이 Markov 모델의 성능보다 우수한 것을 확인할 수 있다. SVR 모델과 RNN 모델의 예측 성능은 거의 비슷하지만, 표 2을 보면 SVR 모델이 유저의 다음 위치를 예측하는데 걸리는 시간이 RNN 모델이 예측하는데 걸리는 시간보다 더 빠르다는 것을 확인할 수 있다. 따라서 이동경로 예측 모델 중 SVR을 통해 구현한 모델이 가장 뛰어난 성능을 가졌다고 할 수 있다.

4.4.3 시뮬레이션 결과

시뮬레이션을 수행할 때는 예측 성능이 가장 좋았던 SVR 모델을 이동경로 예측 모델로 사용하였다. 본 논문에서 제안하는 오프로딩 시스템의 성능을 분석하기 위해 다른 서버로 이동한 후에 DNN 모델을 전송해야 하는 경우(baseline)와 모든 엣지 서버가 DNN 모델을 가지고 있어 모델을 전송하지 않아도 되는 경우(optimal)도 같이 측정하여 비교하였다.

시뮬레이션을 통해 hit ratio와 총 수행된 DNN query의 수를 측정하였다. DNN query는 static query와 dynamic query로 나누어 측정하였다. Static query는 엣지 서버를 이동하지 않고 수행한 DNN query이고, dynamic query는 다른 엣지 서버로 이동한 직후 *time interval*(t) 동안 수행한 DNN query이다. Static query와 dynamic query 중에 이동경로 예측을 통해 증가시킬 수 있는 query는 dynamic query이므로, dynamic query가 본 연구의 optimization target이라고 할 수 있다.

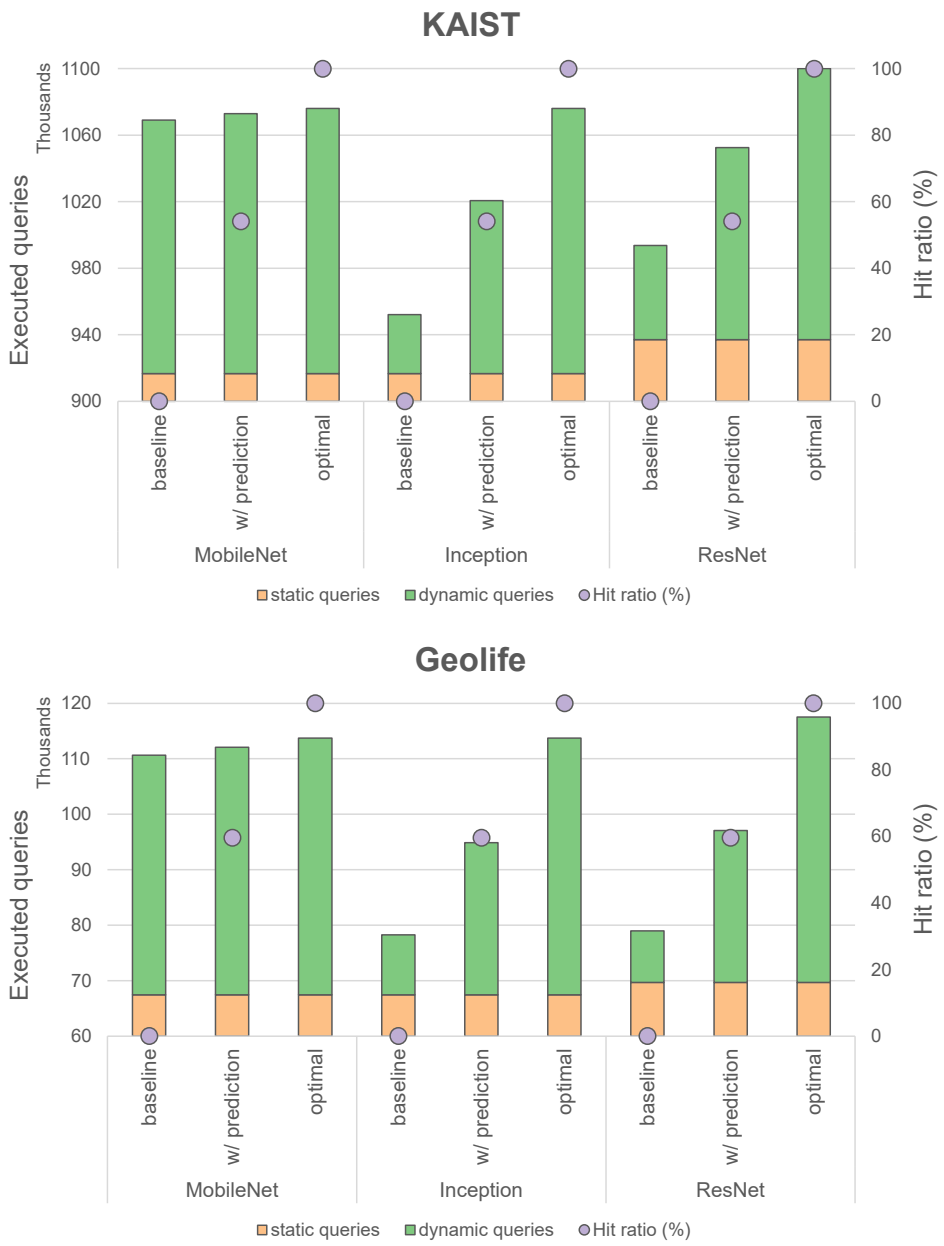

그림 7. 시뮬레이션에서 수행된 query의 수와 hit ratio

그림 7은 시뮬레이션에서 수행된 총 오프로딩 query의 수와 hit ratio를 나타낸다. 모바일 사용자가 다른 서버로 이동하는 경우에만 이동경로 예측을 통한 성능 향상을 얻을 수 있기 때문에 미리 다음 서버를 예측해서 DNN 모델을 전송하면 dynamic query의 수가 증가하게 된다. MobileNet을 DNN 모델로 사용한 어플리케이션의 경우, 모델의 크기가 크지 않기 때문에 전송하는데 시간이 오래 걸리지 않는다. 따라서 이동경로 예측 모델을 통해 다음 서버를 예측하여 모델을 미리 전송해도 성능 향상이 크지 않다. 반면에 Inception과 ResNet의 경우 DNN 모델의 크기가 크기 때문에 이동경로 예측을 통해 모델을 미리 전송함으로써 얻을 수 있는 성능 향상이 크다. KAIST 데이터셋에서는 Inception을 사용한 어플리케이션과 ResNet을 사용한 어플리케이션에서 각각 dynamic query가 baseline에 비해 2.93배, 2.04배 수행되었고, Geolife 데이터셋에서는 각각 2.54배, 2.96배 수행된 것을 확인할 수 있다.

제 5 장 결론

본 논문에서는 모바일 엣지 컴퓨팅 환경에서 모바일 유저의 다음 위치를 예측하여, 예측한 위치와 가까운 엣지 서버로 오프로딩에 필요한 데이터를 미리 전송해 연산 오프로딩 성능을 향상시키는 방법을 제안하였다. Markov Model, Support Vector Regression(SVR), Recurrent Neural Network(RNN)을 사용하여 모바일 유저의 과거 위치를 기반으로 짧은 시간 후(<30초)의 위치를 예측하는 모델을 구현하였다. 구현한 이동경로 예측 모델을 실제 데이터셋에 적용하고, 시뮬레이션을 통해 모바일 엣지 컴퓨팅 환경에서 연산 오프로딩의 성능 향상이 있음을 확인하였다.

참고 문헌

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[3] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pp. 46–57, ACM, 2008.

[4] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive wi-fi mobility data.," in *IEEE INFOCOM*, vol. 2, pp. 1414–1424, INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 2004.

[5] R. Jiang, X. Song, Z. Fan, T. Xia, Q. Chen, S. Miyazawa, and R. Shibasaki, "Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction.," in *AAAI*, pp. 784–791, 2018.

[6] X. Song, H. Kanasugi, and R. Shibasaki, "Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level.," in *IJCAI*, vol. 16, pp. 2618–2624, 2016.

[7] L. Yang, J. Cao, S. Tang, D. Han, and N. Suri, "Run time application repartitioning in dynamic mobile cloud environments," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 336–348, 2016.

[8] D. Ron, Y. Singer, and N. Tishby, "Learning probabilistic automata with variable memory length," in *Proceedings of the seventh annual conference on Computational learning theory*, pp. 35–46, ACM, 1994.

[9] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[10] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, "CRAWDAD dataset ncsu/mobilitymodels (v. 2009-07-23)," jul 2009.

[11] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 312–321, ACM, 2008.

[12] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proceedings of the 18th international conference on World wide web*, pp. 791–800, ACM, 2009.

[13] Y. Zheng, X. Xie, W.-Y. Ma, *et al.*, "Geolife: A collaborative social networking service among user, location and trajectory.," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.

Abstract

Mobility Prediction for Computation Offloading in the Mobile Edge Computing Environment

Hyeonjae Lee
Dept. of Electrical and Computer Engineering
The Graduate School
Seoul National University

Mobile Edge Computing(MEC) paradigm allows mobile devices to offload complex computations to a nearby edge server with low latency. However, in mobile edge computing environment, as a mobile user moves, accessible edge servers constantly change. Moving to another edge server's coverage area cause offloading performance degradation because the server does not have necessary data for offloading.

In this paper, we propose mobility prediction models which predict mobile users' next locations based on their past mobility data and send necessary data for offloading to edge servers close to the predicted locations. Mobility prediction models are implemented by using Markov Model, Support Vector Regression, and Recurrent Neural Network. We evaluated mobility prediction models by performing simulations, which showed that the models can improve computation offloading in mobile edge computing environment.

keywords: Computation Offloading, Edge Computing, Mobility Prediction
Student Number: 2017-28906